# Hybrid Retrieval-Augmented Generation for Real-time Composition Assistance

Menglin Xia\* Xuchao Zhang\* Camille Couturier Guoqing Zheng Saravan Rajmohan Victor Rühle

Microsoft

{mollyxia, xuchaozhang, cacoutur, zheng, saravar, viruh}@microsoft.com

#### **Abstract**

Retrieval augmentation enhances performance of traditional language models by incorporating additional context. However, the computational demands for retrieval augmented large language models (LLMs) pose a challenge when applying them to real-time tasks, such as composition assistance. To address this limitation, we propose the Hybrid Retrieval-Augmented Generation (HybridRAG) framework, a novel approach that efficiently combines a cloud-based LLM with a smaller, clientside, language model through retrieval augmented memory. This integration enables the client model to generate effective responses, benefiting from the LLM's capabilities and contextual information. Additionally, through an asynchronous memory update mechanism, the client model can deliver real-time completions swiftly to user inputs without the need to wait for responses from the cloud. Our experiments on five benchmark datasets demonstrate that HybridRAG significantly improves utility over client-only models while maintaining low latency.

# 1 Introduction

Retrieval-augmented approaches (Lewis et al., 2020; Liu et al., 2022) have emerged as a powerful tool to boost Large Language Model (LLM) performance by incorporating external documents (Lewis et al., 2020; Liu et al., 2022). This integration enables models such as GPT-3 (Brown et al., 2020) and ChatGPT to leverage additional contextual information, resulting in improved contextual understanding and reduced occurrence of hallucinations. However, retrieval-augmented LLMs can be slow and expensive to run due to the size of the model and the extra retrieval step they require, which can cause latency and limit its application in tasks requiring real-time responses, such as composition assistance.

Real-time composition assistance tools are designed to swiftly suggest next words or sentences to help users write faster, and therefore operate within tight latency budgets (typically in the order of 100ms or less). To avoid latency overheads for communicating to the cloud, these models are usually deployed on users' edge devices. This imposes strict constraints on the model's size and capabilities, limiting the effectiveness of composition assistance. While recent advancements have enabled LLMs with 7B parameters (Touvron et al., 2023) to generate 5 tokens per second on a smartphone<sup>1</sup>. they still fall short in terms of achieving real-time responses for composition assistance. In addition, embedding a retrieval-augmentation module into the edge may not always be ideal because relevant documents are often stored in the cloud, such as in companies' cloud storage, and the retrieval process can introduce additional latency overhead.

Hybrid computing between client and cloud models is a promising approach to bridge the gap between the challenges of latency and model performance. However, in existing hybrid computing patterns, such as model routing and split computing (Kudugunta et al., 2021; Matsubara et al., 2022), client and cloud models usually function with synchronized communication. This means that if the cloud model is utilized, the client side must wait for the cloud model to complete its processing before producing the output. Therefore, simply applying existing hybrid patterns to cloud-based LLMs will not resolve the issue of latency and cost.

To address these challenges, we propose a novel Hybrid Retrieval-Augmented Generation (HybridRAG) framework. This framework leverages a cloud model and data stored on the cloud to boost the performance of small language models on client devices through retrieval augmentation, while operating *in an asynchronous manner*.

<sup>\*</sup>These authors contributed equally to this work.

<sup>1</sup>https://news.ycombinator.com/item?id=35171116

The HybridRAG framework consists of a retriever model and an LLM located on the cloud server, as well as an augmentation coordinator and a small model for text prediction deployed on client devices. The client augmentation coordinator sends asynchronous request to the cloud. The cloud retrieves relevant documents and employs an LLM to compress the retrieved documents into shorter snippets of information, which we refer to as *memory*, and sends it asynchronously to the client. On the client side, an instruction-tuned client model leverages available memory to suggest the next words.

The HybridRAG framework offers several benefits. (1) Enhanced utility: HybridRAG enables the client model to make better suggestions by leveraging cloud-based resources. (2) Low latency: our novel hybrid retrieval augmentation framework enables low latency, as asynchronous memory augmentation allows the client model to make predictions without waiting for the cloud. This mitigates the effects of network latency and avoids slow inference inherent to cloud-based retrieval-augmented LLMs. (3) Reduced client-to-cloud communication: the augmentation coordinator minimizes the client-to-cloud communication by requesting augmented memory only when existing memory becomes stale, reducing the frequency of calling the cloud models and thus saving cost. Furthermore, using LLM-compressed memory further reduces data transfer volume.

To evaluate our framework, we conduct experiments on five benchmark datasets from diverse domains. We compare our model to several baselines and show that our client model exhibits substantial utility improvement in text prediction by leveraging cloud-generated memory. In addition, our asynchronous design allowed the framework to operate in real-time and it demonstrated substantial speed improvement compared to a synchronous approach under the same experimental setup. We plan to release our code and data upon publication of the paper under CC BY-NC SA licence.

# 2 Related Work

Hybrid Computing Hybrid computing between edge and cloud devices originated outside the realm of machine learning. It typically divides processing tasks between the edge and the cloud, effectively addressing the limited computation capabilities of edge devices and enabling real-time responses of

critical services (Loghin et al., 2019; Wang et al., 2020). However, literature on hybrid edge-cloud computing for machine learning models is relatively scarce. To our knowledge, the most relevant topic in the literature is split computing, which involves partitioning modules of machine learning pipelines or layers of neural network models between edge and cloud devices to balance overall computation cost and efficiency (Matsubara et al., 2022; Osia et al., 2020). Communication between the edge and the cloud in split computing is inherently synchronized, as both devices contribute to completing one inference run. Another notable paradigm for hybrid computing in machine learning is federated learning, which leverages multiple computing devices for training machine learning models for safety or efficiency purposes (Bonawitz et al., 2019). However, this technique is less commonly used for inference. Cloud service providers such as AWS also have developed patterns for hosting machine learning pipelines across local and cloud devices (AWS-Whitepaper, 2021). The design usually involves splitting the components of a machine learning pipeline, with the core machine learning models still hosted in the cloud. More recently, task-specific model routing (Kudugunta et al., 2021) has also emerged as a promising approach for hybrid computing via routing between client and cloud models. Nonetheless, the overall framework still needs to wait for the cloud model whenever it is used. In addition to hybrid computing, there is also literature on improving efficiency of models deployed on edge devices (Tambe et al., 2021) as well as methods on reducing the size of large models for deployment on smaller devices (Hoefler et al., 2021). These methods are orthogonal to our work.

Retrieval Augmented Models Retrieval augmentation is a technique that enhances a language model with retrieved information from external databases. Various methods have been proposed to integrate the retrieved data into the language model, including the use of prompts (Lewis et al., 2020; Guu et al., 2020; Shi et al., 2023), cross-attention modules (Borgeaud et al., 2021), vector concatenation (Izacard and Grave, 2021; Fan et al., 2021), and output distribution adjustment at decoding (Khandelwal et al., 2020; Liu et al., 2022). In our HybridRAG work, we adopt the prompting method, which incorporates retrieved data into the input.

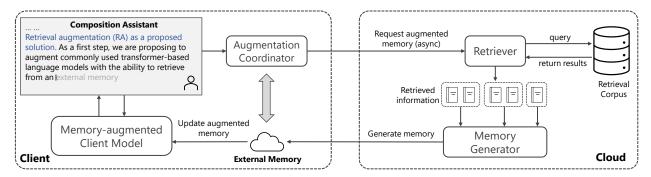


Figure 1: Overview of the HybridRAG framework, which is a hybrid framework for composition assistance. The top left box represents the writing interface. The framework has four main components: augmentation coordinator and client model on the client side (left), and retriever and LLM-based memory generator on the cloud (right).

# 3 Hybrid Retrieval Augmented Generation

We present our HybridRAG framework that leverages cloud-generated memory to enhance the utility of client-based language model while maintaining low latency. The HybridRAG framework consists of four main components: an augmentation coordinator (client), a memory-augmented client model (client), a retriever model (cloud), a LLM-based memory generator (cloud). Figure 1 illustrates the model architecture.

The augmentation coordinator monitors the writing context and determines when to request an augmented memory from the cloud. The retriever model on the cloud then searches the retrieval corpus to find relevant data. Subsequently, the memory generator uses an LLM to construct a memory that includes all essential information from the retrieved data, optimizing its usefulness. Finally, the memory is transmitted to the client and seamlessly integrated into the client model, thereby enhancing its overall performance. Algorithm 1 describes the inference workflow of HybridRAG.

In the following subsections, we discuss the details of the four components.

# 3.1 Augmentation Coordinator

The augmentation coordinator component is responsible for managing the augmented memory  $\mathcal{M}$  by monitoring changes to the writing context. The function of the augmentation coordinator is depicted in Figure 2. To determine whether a memory update is necessary, the coordinator takes into account both the current context  $x_t$  and the context  $x_{t-1}$  from the previous step and calculates the edit distance  $\mathrm{ED}(x_t, x_{t-1})$ . Once the distance exceeds a pre-determined threshold  $\tau$ , the coordinator

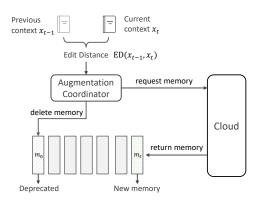


Figure 2: Process of the augmentation coordinator

initiates a request to the cloud server asking for a new memory. We employ the Levenshtein distance (Yujian and Bo, 2007) to measure the token-level difference. To avoid redundant memory requests, we adopt an incremental memory update approach, where only the newly updated context is used as the query input to generate the new memory  $m_t$ . When the augmented memory reaches its maximum capacity of  $\mathcal{M}$ , the earliest stored memory is replaced with the new one. This process is depicted in Figure 2, where we observe that upon reaching the maximum memory capacity, the oldest memory  $m_0$  is swapped out and replaced by the new memory  $m_t$ .

# 3.2 Retrieval-Augmented Memory Generator

Upon receiving a request from the augmentation coordinator, the memory generator on the cloud initiates the preparation of the augmented memory, which will be returned to the client. The memory preparation process consists of two steps: document retrieval and memory generation.

**Document Retrieval** Given an input query x, the goal of the retriever is to select the top-k most rel-

# Algorithm 1: Inference workflow of HybridRAG

```
Data: current user input x_t, input history x_{t-1}, retrieval corpus \mathcal{D}, retrieval model \mathcal{M}_{\text{retrieval}}, cloud-based LLM
         \mathcal{M}_{\mathrm{cloud}}, client model \mathcal{M}_{\mathrm{client}}, memory \mathcal{M}
while x_t do
      ED_t = EditDistance(\boldsymbol{x}_t, \boldsymbol{x}_{t-1});
                                                                                                                  ▷ Send async request to the cloud
      if \mathrm{ED_t} > \tau;
       then
            async \mathcal{D}_r = \{d_1, ...d_k\}: \mathcal{D}_r \sim \mathcal{M}_{\text{retrieval}}(\boldsymbol{x}_t, \mathcal{D});
                                                                                                                 ▷ Retrieve relevant documents
            async m_t \sim \mathcal{M}_{\text{cloud}}(\mathcal{D}_r);
                                                                                                                                     ▷ Generate memory
            \mathcal{M} = Update(\mathcal{M}, m_t);
                                                                                                                                \triangleright Update \mathcal{M} with m_t
            Sample y_t \sim \mathcal{M}_{\text{client}}(x_t, \mathcal{M});
                                                                                               if Accept(\boldsymbol{y}_t) then
                   \boldsymbol{x}_{t-1} \leftarrow \{\boldsymbol{x}_{t-1}, \boldsymbol{x}_t\}, \boldsymbol{x}_t \leftarrow \{\boldsymbol{x}_t, \boldsymbol{y}_t\};
                                                                                                                       \boldsymbol{x}_t \leftarrow \{\boldsymbol{x}_t, Input()\};
                                                                                   ▷ User rejects suggestion and enters new input
            end
      end
end
```

evant documents  $\mathcal{D}_r = \{d_1, \dots, d_k\}$  from a large retrieval corpus  $\mathcal{D}$ , where  $\mathcal{D}_r \subseteq \mathcal{D}$ . Following prior work (Lewis et al., 2020; Ni et al., 2021), we use the Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) method, which is a dual encoder retrieval model pre-trained for question and answering task. DPR encodes the document and the query with a context encoder and a query encoder respectively, and calculates the cosine similarity of the encoded embeddings to retrieve the top-k most relevant documents.

Memory Generation After retrieving the relevant documents  $\mathcal{D}_r$ , instead of directly concatenating them to the original prompt as in Retrieval Augmented Generation (RAG) (Lewis et al., 2020), we employ a LLM to generate concise key takeaways that capture the essential information from the retrieved documents. We hypothesize that the improved representation of the memory could enhance the performance of the client model, which is a small language model that usually struggles with processing long context and has a stricter limit on input context length. In addition, extracting the key takeaways significantly reduces the memory size, resulting in lower communication and inference cost for the client.

To generate concise key takeaways from retrieved documents  $\mathcal{D}_r$ , we first split the documents into text chunks  $\{p_1,\ldots,p_l\}$ , where l is the number of chunks. We choose an appropriate chunk size that maintains sentence integrity, avoiding breaking sentences in the middle. Once the chunks are created, we utilize a LLM to extract the key takeaways from each chunk  $p_i$ . To minimize the frequency of LLM call requests, we consolidate

multiple chunks within one document. We show an example of extracted key takeaways for two text chunks in Appendix A. Subsequently, all the generated bullet points from the retrieval documents are merged to form the memory  $m_t$  for the current t-th memory request. This memory is then combined with the existing memory to construct the new  $\mathcal{M}$  by the augmentation coordinator.

# 3.3 Memory-Augmented Client Model

While most previous work on memory-augmented networks (Weston et al., 2014; Sukhbaatar et al., 2015; Yogatama et al., 2021; Wu et al., 2022) focus on language modeling task performance, our client model aims to effectively leverage the memory generated by the LLM. We hypothesize that although a small language model may lack the capability to handle complex tasks like its larger counterpart, it can still be trained effectively to leverage augmented memory to generate better text completions. To this end, we adopt an instruction-finetuning approach to bolster the client model's ability to effectively leverage cloud-generated memory.

To finetune the client model, we need data triplets of input prompt x, augmented memory  $\mathcal{M}$ , and references for text prediction  $\hat{y}$ . However, obtaining the latter two can be difficult. To address this, we leverage an LLM to generate the necessary training data.

Given a document d, we select a percentage of the document to serve as the input prompt  $x = \mathcal{I}(d)$ . Then we generate the augmented memory  $\mathcal{M}$  with the steps outlined in Section 3.2. As for the reference, a straightforward approach is to directly use the remaining part of the document d. However, this is not ideal since the original text

Prompt	Reference: In 2020, Generative Pre-trained Transformer 3 (GPT-3) was unveiled, a deep learning-based autoregressive language model that can produce human-like text. When provided an initial text as a prompt, it can then generate text that follows on from it This process has eliminated the need for laborious manual labeling and human supervision.  Complete the following text based on the reference: Generative Pre-trained Transformer 3 (GPT-3) is an autoregressive language model released in 2020 that
Output	is capable of producing human-like text when prompted with an initial text. GPT-3 has a 2048-token-long context, a record-breaking 175 billion parameters and a storage capacity of 800GB.

Table 1: Example of constructing an instruction-enhanced prompt for reference generation

may not encompass the information contained in the memory. The discrepancy between the completion and the memory can negatively impact the performance of the client model. To address this issue, we employ the LLM to generate the references. We structure the input prompt and memory into an instruction-based prompt following the format specified in Table 1. The reference can be expressed as  $\hat{y} = \mathcal{M}_{\text{cloud}}(\mathcal{I}(d), \mathcal{M})$ , with  $\mathcal{M}_{\text{cloud}}$ .

After preparing the training data, we proceed to finetune our client model using the instruction-enhanced prompt along with the LLM-generated references. The model is finetuned on the task to predict  $\hat{y}$  given x and  $\mathcal{M}$ . To minimize the discrepancy between our model's predictions y and the LLM-generated references  $\hat{y}$ , we employ the cross-entropy loss on the generated tokens:

$$\mathcal{L}_d = -\sum_{i=1}^{l} \hat{y_i} \log \left( p_{\theta}(y_i | \boldsymbol{x}, \mathcal{M}, \hat{y}_{< i}) \right)$$
 (1)

where l is the length of reference and  $p_{\theta}(\cdot)$  refers to the probability of tokens generated by the client model.

#### 4 Experiments

In this section, we present the experimental results that evaluate the performance of the HybridRAG framework. We introduce the experiment setup in Section 4.1 and present the results of three sets of experiments in Section 4.2. Specifically, Section 4.2.1 evaluates the utility of the proposed method compared against several baselines on multiple benchmark datasets, Section 4.2.2 evaluates the inference latency of our framework, and Section 4.2.3 demonstrates how HybridRAG performs with asynchronous memory updates.

## 4.1 Experimental Setup

**Datasets and Labels** We evaluate our framework on five datasets, including WikiText-103 (Merity et al., 2016), Enron Emails, HackerNews, NIH Ex-Porter, and Youtube Subtitles (Gao et al., 2020).

These datasets encompass a diverse range of domains, including encyclopedia, news, emails, medical documents and video subtitles. For instructiontuning the client model, we use the training set of WikiText-103, which consists of 15,220 Wikipedia passages. We use the first section of each passage to create data for text prediction and retrieve from the remaining sections. To process the training data, we split the first section into sentence chunks between [16, 128] tokens. Each chunk may contain multiple sentences. We then split each chunk into prompt and remaining parts by choosing a random split ratio between [0.125, 0.5]. We evaluate the models on the test splits of the five datasets. During testing, we create a single prompt for each data point from the first section/paragraph for text prediction and retrieve from the remaining sections/paragraphs. We use an LLM to generate references and evaluate the model predictions against LLM-generated references.

**Evaluation Metrics** We employ several automated metrics to evaluate the model's utility. We calculate the perplexity (Jelinek et al., 1977) of the model by measuring how well it predicts the references based on the prompts.

$$PPL = \exp\left(-\frac{1}{l}\sum_{i=1}^{l}log(p_{\theta}(\hat{\boldsymbol{y}}|\boldsymbol{x}))\right) \quad (2)$$

Perplexity indicates the language model's level of uncertainty when processing the given text, with lower perplexity indicating higher model confidence of observing the reference text. In addition, we also use lexical and semantic similarity metrics, GLEU (Wu et al., 2016), BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang et al., 2020), to evaluate the degree of similarity between the model's predictions and the references.

To evaluate the inference latency of our system, we measure the average running time required for three steps in our framework: document retrieval,

		PPL	GLEU	BLEU-4	ROUGE-1	ROUGE-L	METEOR	BERTScore
	Vanilla OPT	9.3	11.4	6.9	27.5	22.1	20.2	84.0
OPT-	RAG	4.3	12.8	9.6	28.4	23.4	22.4	84.5
125M	HybridRAG w/o FT	3.8	14.7	12.2	29.9	25.1	24.3	84.8
	HybridRAG FT	3.4	23.0	21.4	39.6	32.8	34.4	87.0
	HybridRAG IT	2.6	30.2	28.8	48.3	40.2	44.1	89.0
	Vanilla OPT	7.4	13.2	8.8	30.1	24.3	22.8	84.8
OPT-	RAG	3.6	15.4	12.5	31.6	26.0	25.6	85.4
350M	HybridRAG w/o FT	3.3	17.6	15.4	33.5	27.9	28.0	85.7
	HybridRAG FT	3.2	23.9	22.3	40.7	33.8	35.5	87.4
	HybridRAG IT	2.4	32.6	31.4	50.8	42.9	46.6	89.5

Table 2: Peformance comparison of HybridRAG models and baselines on the WikiText-103 dataset

		Enron Emails		NIH ExPorter		Hacker News		Youtube Subtitles	
		PPL	GLEU	PPL	GLEU	PPL	GLEU	PPL	GLEU
OPT- 125M	Vanilla OPT RAG HybridRAG w/o FT HybridRAG FT HybridRAG IT	8.5 6.3 4.6 4.4 3.3	5.8 8.0 9.0 13.8 <b>22.9</b>	7.4 4.4 4.1 3.7 <b>2.9</b>	9.3 10.7 10.9 16.8 <b>24.2</b>	7.5 7.2 5.6 5.3 3.8	8.0 7.5 8.9 14.8 <b>20.2</b>	9.2 7.0 5.9 5.5 <b>4.4</b>	5.7 7.2 7.1 12.5 <b>20.4</b>
OPT- 350M	Vanilla OPT RAG HybridRAG w/o FT HybridRAG FT HybridRAG IT	7.4 5.5 4.1 4.2 <b>3.1</b>	5.9 9.1 12.5 13.3 <b>24.7</b>	6.2 3.7 3.5 3.5 2.7	10.3 12.4 12.6 17.9 <b>25.5</b>	6.4 6.1 4.8 5.1 3.7	8.5 8.4 11.6 13.3 <b>20.7</b>	7.7 5.8 5.0 5.2 <b>4.2</b>	6.3 8.5 9.9 13.4 <b>20.8</b>

Table 3: Performance comparison of HybridRAG models and baselines on four datasets

memory generation, and text prediction. This allows us to quantify the time cost associated with each of these steps and analyze the overall efficiency of our system.

Implementation Details To implement our client model, we compare two OPT language models (Zhang et al., 2022): OPT-125M and OPT-350M. Both models are decoder-only pre-trained transformers that have 125 million and 350 million parameters respectively, which are small enough to run with limited latency budget. We employ greedy search for client model decoding. For the cloud-based LLM, we use the GPT-3.5 textdavinci-003 model from the OpenAI API<sup>2</sup>, and set temperature = 0, top\_p = 1 to make the generation more deterministic. We set the maximum output tokens to 44 for both reference label generation and text prediction. For document retrieval, we process the retrieval text into chunks of 128 tokens and use the DPR model and the Faiss library (Johnson et al., 2019) for efficient retrieval. We set k = 3 after a hyperparameter search.

The client models are trained on machines equipped with one Tesla V100 GPU with 16GB memory. For latency evaluation, we deploy the

client models on two different machines: a GPU machine with an 11GB Nvidia Tesla K80 GPU, and a laptop without a GPU (specifically a Surface 4 laptop featuring an Intel i7 CPU @3.00GHz with 4 cores and 16GB of physical memory). We set the maximum output tokens to 15 for latency evaluation.

**Baseline Methods** We compare our final approach against the following baselines:

- Vanilla OPT: a vanilla client OPT model for text prediction without additional memory or assistance from the cloud.
- RAG: a hybrid setup of the original RAG approach with our framework. In this setting, we retrieve and feed the full retrieved text to the client model without compression. Note that this only works if the documents are sufficiently short to fit in the limited input context of the client model.
- HybridRAG without finetuning (HybridRAG w/o FT) and HybridRAG FT: To assess the efficacy of our instruction-tuned client model, we examine two variants of the client model, one without finetuning and one finetuned to use the memory to predict the original remaining text.

<sup>2</sup>https://platform.openai.com/docs/models/ gnt-3.5

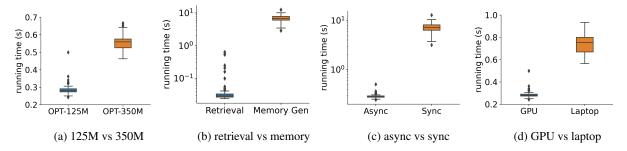


Figure 3: Inference latency for client inference, retrieval and memory generation on multiple devices

When evaluating the baseline models, we ensure a fair comparison by regenerating references for each baseline, based on the memory used by that baseline. Specifically, for the Vanilla OPT baseline, references are generated with LLM without additional memory. For RAG, references are generated with full text.

# 4.2 Experimental Results

# **4.2.1** Utility

Table 2 presents the utility of our models compared to the baselines on the WikiText-103 dataset, with respective results for OPT-125M and OPT-350M models. The results demonstrate that our approach outperforms all client-only and hybrid baselines across the evaluated metrics on WikiText-103. We can observe that the RAG approach outperforms a vanilla client baseline by utilizing retrieved documents as additional context, and the HybridRAG w/o FT model improves upon the RAG approach by using the LLM to extract the key takeaways from the retrieved instead of feeding them directly to the client model. This indicates that the representation of the context is vital to the client model performance. Furthermore, our final model (HybridRAG IT) shows significant improvement over HybridRag w/o FT and HybridRAG FT. This suggests that instruction-tuning helps the model to better leverage the context. Additionally, the OPT-350M based models consistently outperform the OPT-125M based models, suggesting that the size of the model has a significant impact on its overall performance.

Table 3 presents the perplexity and GLEU scores on the other four datasets, with additional metrics provided in Appendix B. Consistent with the findings on the WikiText-103 dataset, our model demonstrates better performance compared to the baseline models across all four datasets. It is worth noting that we did not finetune the client model

specifically on these datasets, which suggests the model's generalization capabilities.

In Appendix C, we provide some examples of working and failing cases of the HybridRAG models.

# 4.2.2 Inference Latency

We performed a latency evaluation for the HybridRAG framework. Figure 3a shows the run times for the client models on a GPU machine. It indicates that the OPT-125M model exhibits a 49.3% faster inference time compared to the OPT-350M model. This emphasizes that the size of the client model plays a crucial role in the inference time. Figure 3b presents the run time for the retrieval and memory generation steps, with the results showing that memory generation utilizing a large language model consumes the majority of the memory preparation time. Figure 3c compares our asynchronous HybridRAG approach with OPT-125M to a synchronous inference approach by directly calling a GPT-3.5 model and a retriever for composition assistance. Notably, our approach showcases an impressive speed enhancement, achieving a remarkable 138.3 times faster performance compared to the synchronous approach. Lastly, we conducted a comparison of the run times of HybridRAG OPT-125M between a GPU machine and a laptop without GPU in Figure 3d. The results indicate that our approach can be deployed on edge devices without GPUs, although the inference time is slower compared to a GPU machine.

It is worth noting that we didn't optimize the client model for decoding speed with caching or quantization. These methods are orthogonal to our work and can be used in conjunction with our approach to further reduce the inference latency.

# 4.2.3 Asynchronous Memory Update

Figure 5 illustrates the impact of asynchronous memory update on model utility. To measure this

Figure 4: An example of setting edit distance threshold = 10 in asynchronous memory update. In this setting, text suggestion is generated from the entire prompt, but only the beginning part of the prompt is used for retrieval and memory generation.

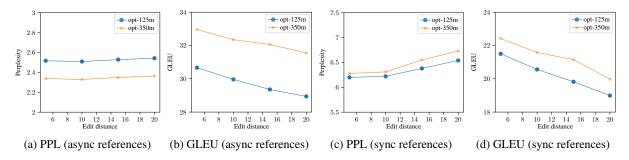


Figure 5: HybridRAG performance with asynchronous memory update. (a) and (b) show the trend of perplexity and GLEU scores with increasing edit distance threshold, evaluated against references generated with the same asynchronous setup. (c) and (d) compare the model predictions against an ideal case of synchronous memory update, where the memory is also generated with the entire prompt.

effect, we conducted an experiment in which we gradually increased the edit distance threshold that determines how often the client model requests for memory updates. Figure 4 shows an example of how we set the edit distance threshold. For each prompt, we use the beginning part of the prompt as the query for retrieval and memory generation and the entire prompt for text prediction, mimicking the case where the memory lags behind the current input context due to asynchronous communication between client and cloud.

Figure 5a and Figure 5b show the trend in perplexity and GLEU scores, with increased edit distance threshold, evaluated against GPT3.5 generated references with the same asynchronous setup. They show that model utility remains relatively stable in perplexity with a deceasing trend in GLEU compared to LLMs. Figure 5c and Figure 5d show the perplexity and GLEU scores of the client model under the asynchronous setup, evaluated against references generated in an ideal synchronous memory update setup, where the memory is also created using the entire prompt without lag. Due to the difference in the freshness of the memory, there is a larger gap between the asynchronous predictions and the synchronous references. As the edit distance threshold increases, the memory becomes less up-to-date due to the increased difference between the query used for memory generation and current input for text prediction, resulting in a decline in model utility. Nevertheless, it still significantly outperformed the vanilla OPT baselines.

# 5 Conclusion

In this paper, we propose HybridRAG, a novel hybrid retrieval-augmented generation approach for real-time composition assistance. By integrating LLM-enhanced memory into our instruction-tuned client model with asynchronous update, we show with experiment results on multiple datasets that our hybrid retrieval approach enables substantial utility improvements over smaller language models while maintaining inference efficiency, making it a valuable solution for real-time tasks.

In our work, we employ retrieval-based memory augmentation as the solution to combine the powerful LLM on the cloud and the more agile client model. Naturally, the performance of the system relies on the quality of the memory that the cloud provides to the client and how the memory is integrated into the client model. The quality of the memory is influenced by multiple factors: the representation of the memory, the relevance of the retrieved data, and the freshness of information compared to the current input context. In future work, we will continue to investigate more effective ways of refining memory and explore alternative memory augmentation approaches to further enhance our model performance.

#### Limitations

To better understand the strengths and limitations of HybridRAG, we manually examined the completions of different models. We find that the performance of the client model highly depends on the memory. For example, as the memory becomes less up-to-date with increased edit distance threshold in asynchronous memory update, the performance of the client model inevitably declines. Additionally, we have observed cases where the client model combines information from different bits of the memories, resulting in fabrication of inaccurate information. In addition, while an LLM can ignore the memory and use its parametric knowledge for generation when the augmented memory deviates from the input prompt, the smaller client model tends to adhere to the memory content. Improving the memory generator by reducing duplicate information, and enhancing the reasoning abilities of the client model or encouraging it to effectively select and ignore memory content would be some of the directions to address these failing cases and limitations.

# **Ethical Considerations**

HybridRAG is a composition assistance tool that integrates client and cloud models and data. In our implementation, data is transmitted between the client and cloud as plain text. However, this transmission process poses potential privacy and confidentiality risks. To mitigate these risks, security measures such as cryptography and access controls can be implemented.

#### References

AWS-Whitepaper. 2021. Hybrid machine learning.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2021. Improving language models by retrieving from trillions of tokens. *CoRR*, abs/2112.04426.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2021. Augmenting transformers with KNNbased composite memory for dialog. *Transactions of* the Association for Computational Linguistics, 9:82– 99

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(1)

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

- Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.
- Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3577–3599, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Ruibo Liu, Guoqing Zheng, Shashank Gupta, Radhika Gaonkar, Chongyang Gao, Soroush Vosoughi, Milad Shokouhi, and Ahmed Hassan Awadallah. 2022. Knowledge infused decoding. *arXiv preprint arXiv:2204.03084*.
- Dumitrel Loghin, Lavanya Ramapantulu, and Yong Meng Teo. 2019. Towards analyzing the performance of hybrid edge-cloud processing. In 2019 IEEE International Conference on Edge Computing (EDGE), pages 87–94.
- Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Comput. Surv.*, 55(5).
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models
- Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.
- Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R. Rabiee, Nicholas D. Lane, and Hamed Haddadi. 2020. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, 7(5):4505–4518.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

- Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. Replug: Retrieval-augmented black-box language models.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. *Advances in neural information processing systems*, 28.
- Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. 2021. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 830–844, New York, NY, USA. Association for Computing Machinery.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint* arXiv:2302.13971.
- Bo Wang, Changhai Wang, Wanwei Huang, Ying Song, and Xiaoyun Qin. 2020. A survey and taxonomy on task offloading for edge-cloud computing. *IEEE Access*, 8:186080–186101.
- Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.
- Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.
- Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373.
- Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

# A Example of Memory Generation

Table 4 shows an example of augmented memory generated by GPT-3.5.

# **B** More results on utility evaluation

The results of the model utility on Enron Emails, NIH ExPorter, HackerNews, and YouTubeSubtitles datasets evaluated in all seven metrics are presented in Tables 5 and 6. We can observe that our model consistently outperforms all the other baselines.

# C Examples of the model completions

Table 7 shows a working example for HybridRAG models and Table 8 and 9 show examples of failing cases

Table 8 is a failing case for both OPT-125M and OPT-350M HybridRAG models. In this case, the memory doesn't contain the information needed to complete the text. As a large language model, GPT3.5 is capable of ignoring the memory and using its parametric memory to generate the completion. However, the smaller client models tend to pick entities present in the memory for text generation despite that the resulting completion is not factually accurate. Table 9 shows an example of working case for HybridRAG OPT-350M IT model, but a failing case for the OPT-125M based model. In this case, the memories are bullet points generated from several document chunks; the client model with limited reasoning abilities does not allow them to process the memories extensively and reorganize them. We've noticed that when the small OPT client models combine information from different bits of the memories, they usually end up fabricating wrong facts.

Improving the memory generator by reducing duplicate information, and enhancing the reasoning abilities of the client model or encouraging it to select more useful memories and ignore irrelevant content would be some of the ways to address these failing cases and limitations.

# Prompt Your task is to carefully read each paragraph and generate a list of key takeaways from the paragraphs in concise sentences. Key takeaways for each paragraph should be no longer than 64 words and should include important details such as facts, entities, persons, organizations, numbers, years, and locations. Please keep each key takeaway short. When referring to previously mentioned entities, use the entity name instead of the pronoun 'it'. Please ensure that your output adheres to these guidelines to the best of your ability. P1: The pier projects from the seafront at Clevedon into the Severn Estuary, which separates South West England from South Wales. The pier and toll house, where entry fees are collected, are adjacent to the Royal Pier Hotel, originally known as The Rock House and built in 1823 by Thomas Hollyman. The shore at Clevedon is a mixture of pebbled beaches and low rocky cliffs, with the old harbour being at the western edge of the town at the mouth of the Land Yeo river. The rocky beach has been designated as the Clevedon Shore geological Site of Special Scientific Interest. P2: The pier was dismantled in 1985, taken to Portishead dock for restoration, and reconstructed in 1986. After a long campaign by local people to raise funds for restoration (supported by Sir John Betjeman, who described Clevedon as "the most beautiful pier in England"), the pier eventually reopened. Some funds were raised by "sponsored planks" - small brass plaques with names or messages are inlaid on the wooden planks and benches, recording donations. Reconstruction of the pier spans and decking was completed on 27 May 1989, and the pier was reopened to great enthusiasm. Key Takeaways: ### P1: Output - The pier is located in Clevedon, South West England, at the mouth of the Land Yeo river. - The shore at Clevedon is a mixture of pebbled beaches and low rocky cliffs.

Table 4: Example of augmented memory generated by GPT-3.5.

- A long campaign was launched to raise funds for restoration, supported by Sir John Betjeman. ...

- The pier was dismantled in 1985 and reconstructed in 1986.

		PPL	GLEU	BLEU-4	ROUGE-1	ROUGE-L	METEOR	BERTScore
	Vanilla OPT	7.4	5.9	2.7	17.3	14.3	13.2	80.2
Enron	RAG	5.5	9.1	6.6	21.7	18.1	17.0	80.6
Emails	HybridRAG w/o FT	4.1	12.5	10.8	25.3	21.6	21.1	81.8
Emans	HybridRAG FT	4.2	13.3	11.6	26.5	22.1	22.8	83.1
	HybridRAG IT	3.1	24.7	22.7	43.9	35.4	39.6	87.9
	Vanilla OPT	6.2	10.3	5.4	27.7	22.3	19.6	85.3
NIH	RAG	3.7	12.4	8.9	30.2	24.5	23.3	85.8
ExPorter	HybridRAG w/o FT	3.5	12.6	9.3	30.0	24.6	23.7	85.7
EXPORTE	HybridRAG FT	3.5	17.9	15.4	36.5	29.4	30.6	87.2
	HybridRAG IT	2.7	25.5	23.2	45.9	37.2	41.2	89.2
	Vanilla OPT	6.4	8.5	5.0	24.7	20.5	16.3	84.9
Hacker	RAG	6.1	8.4	5.6	22.4	18.9	14.7	83.9
News	HybridRAG w/o FT	4.8	11.6	9.2	27.0	22.6	19.4	84.9
news	HybridRAG FT	5.1	13.3	11.4	28.2	23.0	21.6	84.8
	HybridRAG IT	3.7	20.7	18.2	40.3	31.6	35.3	87.8
	Vanilla OPT	7.7	6.3	2.7	17.8	15.1	13.8	82.2
V1	RAG	5.8	8.5	5.2	22.3	18.1	17.4	83.5
Youtube	HybridRAG w/o FT	5.0	9.9	7.4	22.1	18.4	18.1	83.2
Subtitles	HybridRAG FT	5.2	13.4	11.0	27.1	22.0	23.0	84.5
	HybridRAG IT	4.2	20.8	18.3	39.2	30.7	34.7	87.4

Table 5: Comparison of the utility performance of the OPT-350M-based HybridRAG models and baselines on four datasets

		PPL	GLEU	BLEU-4	ROUGE-1	ROUGE-L	METEOR	BERTScore
	Vanilla OPT	8.5	5.8	2.6	17.4	14.7	13.5	80.1
Enron	RAG	6.3	8.0	5.9	20.0	17.1	15.4	79.6
Emails	HybridRAG w/o FT	4.6	9.0	6.9	20.8	17.9	16.9	80.9
Ellialis	HybridRAG FT	4.4	13.8	12.1	26.9	22.6	23.3	83.3
	HybridRAG IT	3.3	22.9	20.9	41.6	33.3	37.1	86.9
	Vanilla OPT	7.4	9.3	4.5	25.9	21.1	18.3	84.8
NIH	RAG	4.4	10.7	7.1	27.4	22.5	20.8	84.9
ExPorter	HybridRAG w/o FT	4.1	10.9	7.7	26.9	22.5	21.0	84.9
EXPORTE	HybridRAG FT	3.7	16.8	14.4	34.9	28.3	29.3	86.7
	HybridRAG IT	2.9	24.2	21.9	44.3	35.6	39.4	88.8
	Vanilla OPT	7.5	8.0	4.6	23.1	19.4	15.3	84.1
Hacker	RAG	7.2	7.5	4.8	20.9	18.0	13.4	83.4
News	HybridRAG w/o FT	5.6	8.9	6.4	22.6	19.4	15.3	83.8
news	HybridRAG FT	5.3	14.8	12.8	30.3	24.8	23.6	85.4
	HybridRAG IT	3.8	20.2	18.0	39.3	30.8	33.3	87.5
	Vanilla OPT	9.2	5.7	2.2	16.7	14.2	13.1	82.6
Youtube	RAG	7.0	7.2	4.1	19.4	16.7	15.5	82.9
Subtitles	HybridRAG w/o FT	5.9	7.1	4.0	18.2	15.7	14.8	82.1
Subilles	HybridRAG FT	5.5	12.5	9.9	26.1	21.4	22.5	84.6
	HybridRAG IT	4.4	20.4	17.8	38.7	30.5	34.8	87.3

Table 6: Comparison of the utility performance of the OPT-125M-based HybridRAG models and baselines on four datasets

prompt	Air Mata Iboe (Perfected spelling: Air
GPT-3.5 generated reference	Mata Ibu) was a musical extravaganza released in December 1941. It was distributed by Columbia Pictures and premiered at Sampoerna Theatre in Surabaya. It was advertised under the Dutch title Tran
HybridRAG OPT-125M IT	Mata Iboe) was released in December 1941 and was distributed by Columbia Pictures. It was advertised under the Dutch title Tranen Eener Moeder and was promoted as a "musical extravaganza".
HybridRAG OPT-350M FT	Mata Iboe) was released in December 1941. It was distributed by Columbia Pictures and premiered at Sampoerna Theatre in Surabaya. It was advertised under the Dutch title Tranen Eener Moeder
HybridRAG OPT-350M IT	Mata Iboe) was released in December 1941 and was distributed by Columbia Pictures. It was advertised under the Dutch title Tranen Eener Moeder and was promoted as a "musical extravaganza".

Table 7: A working case for HybridRAG models

prompt	Viva el Príncipe (English: Long Live the Prince) is the thirteenth studio album by Mexican recording artist
GPT-3.5 generated reference	Cristian Castro. It was released on November 30, 2010 and was met with critical acclaim. The album peaked at No. 5 on the Argentine Chamber of Phonograms and Videograms Producers pop charts and received gold
HybridRAG OPT-125M IT	David Jeffries. It was released on November 30, 2010 worldwide and peaked at No. 5 on the Argentine Chamber of Phonograms and Videograms Producers pop charts. The album was certified diamond by AMPROF
HybridRAG OPT-350M FT	David Jeffries. It was released on November 30, 2010 worldwide. In Mexico, it spent fifteen weeks on top of the Top 100 Mexico chart. It was certified diamond by AMPROFON for shipping 300,000
HybridRAG OPT-350M IT	David Jeffries. It was released on November 30, 2010 worldwide and peaked at No. 5 on the Argentine Chamber of Phonograms and Videograms Producers pop charts. It was certified diamond by AMPROFON

Table 8: A failing case for HybridRAG models.

prompt	"Make Me Like You" is a song by American singer Gwen Stefani
GPT-3.5 generated reference	, released as the second single from her third studio album This Is What the Truth Feels Like. The track was written by Stefani, Justin Tranter, Julia Michaels, Mattias Larsson and Robin Fredriks
HybridRAG OPT-125M IT	that has become a classic. Released digitally on February 12, 2016 as the album's second single, it was serviced to mainstream radio on February 16, 2016 in the United States. The track was written by Stefani
HybridRAG OPT-350M FT	, and was released digitally on February 12, 2016 as the album's second single. It was serviced to mainstream radio on February 16, 2016 in the United States. Stefani wrote the song with her husband, fellow
HybridRAG OPT-350M IT	that was released digitally on February 12, 2016. It was written by Stefani, Justin Tranter, Julia Michaels, Mattias Larsson and Robin Fredriksson under their stage name Mattman & Robin. The

Table 9: A working case for HybridRAG OPT-350M IT but failed for other variants.